Internship report

Parametric estimation of response times

An adapted EM algorithm for fixed-priority scheduling policies

Author:
Olena Verbytska

Supervisor:
Kevin Zagalo

August 31, 2022

Contents

1 Kopernic team

1.1 Overview

A cyber-physical system (CPS) has cyber (or computational) components and physical components that communicate. The Kopernic team deals with the problem of studying time properties (execution time of a program or the schedulability of communicating programs, etc.) of the cyber components of a CPS. The cyber components may have functions with different criticalities with respect to time and a solution should come with associated proofs of its appropriateness for each criticality. A solution is appropriate for a criticality level if all functions fulfill the expectations of that criticality level.

Based on their mathematical foundations, the solutions are: either non-probabilistic when all time properties are estimated and/or bounded by numerical values or probabilistic when at least one time property is estimated and/or bounded by probability distributions.

The Kopernic team proposes a system-oriented solution to the problem of studying time properties of the cyber components of a CPS. The solution is expected to be obtained by composing probabilistic and non-probabilistic approaches for these systems.

1.2 Research directions

- Proposing a classification of variability factors of execution times for a program with respect to the processor features.
- Defining a compositional rule of statistical models based on Bayesian approaches for bounds on the execution times of programs.
- Building scheduling algorithms taking into account the interaction between different variability factors.
- Proving schedulability analyses for the proposed scheduling algorithms.
- Deciding the schedulability of programs communicating through predictable and non-predictable networks.

2 Real-time systems

Real-time systems is a specific field in computer science. Usually used for embedded systems with small energy and computing resources, they have a specific design with a micro-controller architecture and a set of programs (or *tasks*) running on it. An important part of this design, is to make correspond a processing unit (a CPU for example) neither over nor under dimensioned for a given set of programs. This set must be by construction *schedulable*, meaning that there must exist an order (or a *schedule*) in which programs should execute in a way

that they respect timing requirements that we call *deadlines* in the given processing unit.

In this work we consider *periodic* tasks, meaning that an instance of each task is periodically released at a given rate, and a single-core system, *i.e.* only one task is processed at a time. The studied schedule is called the *fixed-priority scheduling policy*, meaning that each task has its own priority and that if a task is released it preempts all tasks with a lower priority and execute until it is finished or preempted by a higher priority task. The deadlines are *implicit*, meaning that an instance of a task should be over before the next instance of the same task in order to respect its requirements.

2.1 Model

Let us consider a single-core processor real-time system composed of a finite task set $\Gamma = \{\tau_1, \ldots, \tau_N\}$ ordered by decreasing priority order and scheduled with a fixed-priority preemptive policy, which means that the task τ_i will stop (*i.e.* preempt) any running task τ_j with $j > i$ to execute itself.

A task τ_i is characterized by:

- its *execution time* $C_i > 0$, with a distribution function F_i, and finite mean m_i and standard deviation s_i,
- its *inter-arrival time* $T_i > 0$ between two consecutive instances of τ_i, with a distribution function G_i of rate $\lambda_i \triangleq 1/\mathbb{E}[T_i]$,
- its *relative implicit deadline* $D_i > 0$, with the same distribution function G_i as its inter-arrival time.

The j-th instance of the task τ_i is called a *job* and we denote it $\tau_{i,j}$. Its execution time is denoted $C_{i,j}$ and with distribution function F_i. A job is said to be a job of *level* i if any job of a task has an higher or equal priority than τ_i, *i.e.* any job $\tau_{k,j}, 1 \leq k \leq i$, $j \in \mathbb{N}$. $T_{i,j}$ is the *inter-arrival time* between $\tau_{i,j-1}$ and $\tau_{i,j}$, with distribution function G_i. Let the *mean utilization of level* i be $u_i = \sum_{j=1}^{i} \lambda_j m_j$ and the *deviation of level* i be $v_i = (\sum_{j=1}^{i} \lambda_j s_j)^{1/2}$.

The *response time* $R_{i,j}$ of a job $\tau_{i,j}$ is the elapsed time between its arrival time $A_{i,j} \triangleq \sum_{k=1}^{j} T_{i,k}$ and the end of its execution. We consider implicit deadlines, *i.e.* the absolute deadline of the job $\tau_{i,j}$ is $A_{k,\ell+1}$ and its relative deadline is $A_{i,j+1} - A_{i,j} = T_{i,j+1} \sim D_i$ and with distribution function G_i. For the sake of comprehension, we refer to the relative deadline of τ_i with the variable D_i and to its inter-arrival time T_i, even though they have the same distribution. At each arrival of jobs of a same task, previous jobs miss their deadline and are discarded. Furthermore, note that all the results presented in this work hold while execution times, periods and deadlines (C_i, T_i, D_i) are mutually dependent.

A job $\tau_{i,j}$ respects its deadline if its response time $R_{i,j}$ is lower or equal to its deadline, *i.e.* $R_{i,j} \leq D_k$. In order for a job to be *schedulable*, its *deadline failure probability* $\mathbb{P}(R_{i,j} > D_k)$ should be lower than its *permitted failure rate*

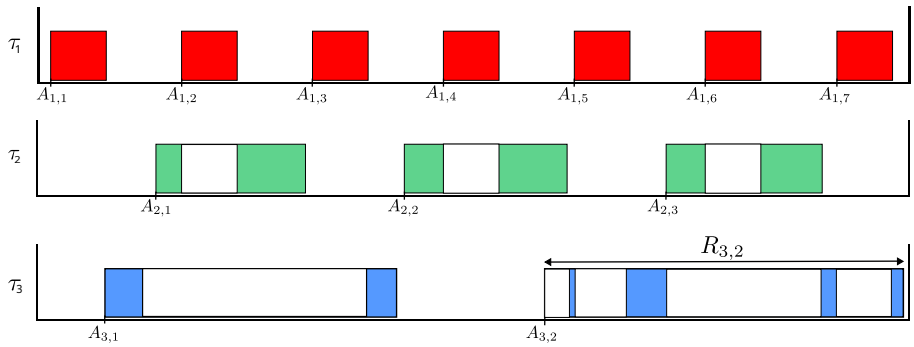


Figure 1: Example of a fixed-priority schedule.

$\alpha_k \in (0,1)$, *i.e.*

$$\mathbb{P}\left(R_{i,j} > D_k\right) = \int \mathbb{P}\left(R_{i,j} > t\right) dG_k(t) \leq \alpha_k \tag{1}$$

When a job does not meet its timing requirements, it is immediately discarded without overhead. This discarding policy is arbitrary, and simplifies the analysis of response times. It allows to express response times as functions of their associated execution times and the higher priority jobs execution times. It is also the reason why response times of jobs of a same task are independent.

2.2 Heavy-traffic response times

In real-time scheduling, the systems need to be by construction such that no job misses its deadline. In order to do so, the task sets are built for the worst-case scenario, *i.e.* the context providing the largest response time has to satisfy the timing constraints. Following Liu and Layland [6], Zagalo *et al.* proved that in the context of implicit deadlines, when $u_i < 1$, the blocking time process is stationary. They introduce the *heavy-traffic response time*

$$\tilde{R}_i = \inf\left\{t > 0 : \hat{W}_{i-1}(t) = t - C_i - \tilde{\beta}_{i-1}\right\} \tag{2}$$

where $\hat{W}_i$ is a Brownian motion of drift u_i and deviation v_i and $\tilde{\beta}_i$ is the positive and finite variable. In order to build a bound of response times, we assume that the processor is always at its minimum speed of 1.

From (2), response times are expressed as *first-passage time* of Brownian motions, which led to the following theorem.

Theorem 2.2.1 (Zagalo *et al.* [9])**.** *If $u_i < 1$, the distribution function of the heavy-traffic steady-state response time R_i of the task τ_i is*

$$h_i(r) = \int \varphi\left(r; \frac{\theta}{1-u_{i-1}}, \left(\frac{\theta}{v_{i-1}}\right)^2\right) d\pi_i(\theta) \tag{3}$$

where π_i is an unknown distribution function and φ is probability density function of inverse Gaussian distribution.

The estimation is better when i is large and when the support of the functions $F_j, j = 1, \ldots, i$ are wide. However, as this representation of response times is accurate, we need to estimate $d\pi_i(\theta)$.

3 Finite mixture models

3.1 Definition

Let $y_j \in \mathbb{R}^d$, $j = \overline{1,n}$ be multivariate observations. A finite mixture model represents the probability density function of one multivariate observation, y_j, as a weighted average of m probability density functions, called *mixture components*:

$$h(y_j) = \sum_{k=1}^{m} \pi_k f_k(y_j; \theta_k), \tag{4}$$

where π_k is the probability that an observation was generated by the k-th component, with $\pi_k \geq 0$ for $k = \overline{1,m}$, and $\sum_{k=1}^{m} \pi_k = 1$, while $f_k(\cdot; \theta_k)$ is the density of the k-th component given the values of its parameters θ_k.

3.2 EM algorithm for finite mixture models

The most common way to estimate parameters π_k and θ_k, $k = \overline{1,m}$ by maximum likelihood is via the Expectation-Maximization (EM) algorithm [7]. Let us consider the data as n multivariate observations (y_j, z_j), in which y_j is observed and z_j is unobserved. If the (y_j, z_j) are independent and identically distributed according to pdf f with parameters θ, then the *complete-data likelihood* is

$$L_c(y, z; \theta) = \prod_{j=1}^{n} f(y_j, z_j; \theta), \tag{5}$$

where $y = (y_1, ..., y_n)$ and $z = (z_1, ..., z_n)$, with $z_j = (z_{j,1}, ..., z_{j,m})$, $j = \overline{1,n}$ such that

$$z_{j,k} = \begin{cases} 1, \text{if } y_j \text{ belongs to group } k, \\ 0, \text{otherwise.} \end{cases} \tag{6}$$

We assume that the z_j are independent and identically distributed, each according to a multinomial distribution of one draw from m categories with probabilities $\pi_1, ..., \pi_m$. We also assume that the density of an observation y_j given z_j is given by $\prod_{k=1}^{m} (\pi_k f_k(y_j; \theta_k))^{z_{j,k}}$. Then the resulting complete-data log-likelihood is

$$l_c(\theta, \pi_k, z; y) = \sum_{j=1}^{n} \sum_{k=1}^{m} z_{j,k} \log[\pi_k f_k(y_j; \theta_k)]. \tag{7}$$

The EM algorithm alternates between two steps. The first is an *expectation step* (E-step), in which the conditional expectation of the complete data log-likelihood given the observed data and the current parameter estimates is computed. The second is a *maximization step* (M-step), in which the parameters that maximize the expected log-likelihood from the E-step are determined.

The s-th iteration of the EM algorithm is given by

- E-step:

$$\hat{z}_{j,k}^{(s)} = \frac{\hat{\pi}_k^{(s-1)} f_k(y_j; \hat{\theta}_k^{(s-1)})}{\sum_{h=1}^{m} \hat{\pi}_h^{(s-1)} f_h(y_j; \hat{\theta}_h^{(s-1)})}. \tag{8}$$

- M-step:

$$\hat{\pi}_k^{(s)} = \frac{\sum_{j=1}^{n} \hat{z}_{j,k}^{(s-1)}}{n}, \tag{9}$$

$$\hat{\theta}^{(s)} = \text{argmax}_\theta \, l_c(\theta, \hat{\pi}^{(s)}, \hat{z}^{(s)}; y), \tag{10}$$

$$l_c(\theta, \hat{\pi}^{(s)}, \hat{z}^{(s)}; y) = \sum_{j=1}^{n} \sum_{k=1}^{m} \hat{z}_{j,k}^{(s)} \log[\hat{\pi}_k^{(s)} f_k(y_j; \theta_k)]. \tag{11}$$

It was shown in [7, 5] that EM algorithm converges to a local maximum of the log-likelihood function under certain mild conditions. The EM algorithm can converge very slowly in case of a poor choice of initial values. Therefore, we will use the results of k-means algorithm to initialize the EM algorithm. In practice it is said that the algorithm is converged, if it starts moving slowly in the latest iterations. Possible criteria are that the log-likelihood or the model parameters have changed very little between the last two iterations; a typical threshold is a change of less than 10^{-5}.

3.3 Choosing the number of clusters

One approach to the problem of model selection in clustering is based on Bayesian model selection [3]. The main idea is that if several statistical models with different number of clusters, $M_1, ..., M_K$ are considered, with prior probabilities $p(M_k), k = \overline{1, K}$, then by Bayes' theorem the posterior probability of model M_k given data D is proportional to the probability of data given model M_k times the model's prior probability:

$$p(M_k \mid D) \propto p(D \mid M_k) p(M_k) \tag{12}$$

When there are unknown parameters, then by the law of total probability $p(D \mid M_k)$ is obtained by integrating over the parameters, i.e.

$$p(D \mid M_k) = \int p(D \mid \theta_{M_k}, M_k) p(\theta_{M_k} \mid M_k) d\theta_{M_k}, \tag{13}$$

where $p(\theta_{M_k} \mid M_k)$ is prior distribution of θ_{M_k}, the parameter vector for model M_k.

$p(D \mid M_k)$ is called *marginal likelihood* of model M_k. Marginal likelihood can be approximated by the Bayesian Information Criterion (BIC)

$$2 \log p(D \mid M_k) \approx 2 \log p(D \mid \hat{\theta}_{M_k}, M_k) - \nu_{M_k} \log(n) = BIC_{M_k}, \qquad (14)$$

where ν_{M_k} is a number of independent parameters to be estimated in model M_k [4].

4 Re-parametrized inverse Gaussian distribution for response times

As the representation in (3) is helpful to real-time designers, the parameters both depend on the blocking time distribution. In order to make a better estimation, and less expensive tests, we apply in this section a re-parametrization of the inverse Gaussian distribution suited.

In [1], the author introduces a modified version of the inverse Gaussian distribution, using its mode μ and smoothing parameter γ instead of the mean ξ and shape λ. This reparametrized inverse Gaussian distribution (rIG) of parameters (μ, γ) is defined by its probability density function

$$\phi(r; \mu, \gamma) = \sqrt{\frac{\mu(3\gamma + \mu)}{2\pi\gamma r^3}} \exp\left\{ -\frac{\left(r - \sqrt{\mu(3\gamma + \mu)}\right)^2}{2\gamma r} \right\}$$

so that $\phi(r; \mu, \gamma) = \varphi(r; \xi, \lambda)$ when $\mu = \xi \left(\sqrt{1 + \frac{9\xi^2}{4\lambda^2}} - \frac{3\xi}{2\lambda} \right)$ and $\gamma = \xi^2/\lambda$.

Its maximum likelihood estimator is found by the *Expectation-Maximization* (EM) algorithm described in [1]. The mode of the inverse Gaussian distribution of parameters $\theta/(1-\bar{u}_i),\ \theta^2/v_i^2$ is associated to the rIG distribution parameters

$$\mu_i(\theta) = \sqrt{\left(\frac{\theta}{1-u_i}\right)^2 + \frac{9\gamma_i^2}{4}} - \frac{3\gamma_i}{2}$$

and γ_i is the *smooth parameter* defined by

$$\gamma_i = \frac{v_i^2}{(1-u_i)^2} \qquad (15)$$

and does not depend on θ.

Let us set the probability density function of parameter $\theta > 0$,

$$\psi_i(r; \theta) \triangleq \phi(r; \mu_i(\theta), \gamma_i)$$

as the Response time-inverse Gaussian (RT-IG) distribution.

Corollary 4.0.1. *If $u_i < 1$, the probability density function of the heavy-traffic response time R_i of the task τ_i is*

$$h_i(r) = \int \psi_{i-1}(r;\theta) d\pi_i(\theta) \tag{16}$$

When the functions $(F_i)_i$ are discrete and the support of π_i is small enough, the exact values of h_i are easy to find. We find in the next section the appropriate parametric estimation and non-parametric estimation of the probability density function h_i in the cases where the $(\gamma_i)_i$ are known.

4.1 Approximation of heavy-traffic response times

Let us focus on a task $\tau_i \in \Gamma$. The empirical observation over many data sets of response times, *e.g.*[8], justifies the need for not only an estimation of a distribution but a mixture of this distribution. This means that distributions, the inverse Gaussian one in our case, with different parameters are joint together, describing M different type of behaviors of the system. An interpretation of these mixtures is the integration in (3). Formally, this means that we approximate the probability density function of the response time R_i with a probability density function

$$h_i(r;\boldsymbol{\pi}_i,\boldsymbol{\theta}_i) = \sum_{k=1}^{m_i} \pi_{i,k} \psi_{i-1}\left(r;\theta_{i,k}\right) \tag{17}$$

In real-time systems, the interest of such approach is to measure a quality of service (QoS) with a given probability of failure. For example, a task τ_i should not miss its deadline with a permitted failure rate α_i, then the inequality in (1) is approximated with the mixture (17).

4.2 Maximum likelihood estimation of response times

In this section we present the maximum likelihood estimator (MLE) proposed by [1] $(\boldsymbol{\pi}_i, \boldsymbol{\theta}_i)$. The complete-likelihood can be written as

$$L_c(\boldsymbol{Z}_i, \boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \prod_{j=1}^{n} \prod_{k=1}^{m_i} [\pi_{i,k} \psi_{i-1}(r_j;\theta_{i,k})]^{Z_{i,j,k}}$$

and the complete log-likelihood $\ell_c = \log L_c$ is

$$\ell_c(\boldsymbol{Z}_i, \boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \ell_{c_1}(\boldsymbol{Z}_i, \boldsymbol{\pi}_i) + \ell_{c_2}(\boldsymbol{Z}_i, \boldsymbol{\theta}_i)$$

where

$$\ell_{c_1}(\boldsymbol{Z}_i, \boldsymbol{\pi}_i) = \sum_{j=1}^{n} \sum_{k=1}^{m_i} Z_{i,j,k} \log \pi_{i,k}$$

and

$$\ell_{c_2}(\boldsymbol{Z}_i, \boldsymbol{\theta}_i) = \sum_{j=1}^{n} \sum_{k=1}^{m_i} Z_{i,j,k} \log \psi_{i-1}(r_j;\theta_{i,k})$$

which leads to the following EM algorithm:

E-step For the $(s+1)$-th step of the EM algorithm, $\boldsymbol{z}_i^{(s)}$ the conditional expectation of $\boldsymbol{Z}_i$ given $(\boldsymbol{\pi}_i, \boldsymbol{\theta}_i) = \left(\boldsymbol{\pi}_i^{(s)}, \boldsymbol{\theta}_i^{(s)}\right)$ is given by

$$z_{i,j,k}^{(s)} = \frac{\pi_{i,k}^{(s)} \psi_{i-1}\left(r_j; \theta_{i,k}^{(s)}\right)}{h_i\left(r_j; \boldsymbol{\pi}_i^{(s)}, \boldsymbol{\theta}_i^{(s)}\right)} \tag{18}$$

M-step For the $(s+1)$-th step of the EM algorithm, $\ell_{c_1}(\boldsymbol{z}_i^{(s)}, \cdot)$ is maximized by

$$\pi_{i,k}^{(s+1)} = \frac{1}{n} \sum_{j=1}^{n} z_{i,j,k}^{(s)},\ k = 1, \ldots, m_i \tag{19}$$

and maximizing ℓ_{c_2} with respect to $\boldsymbol{\theta}$ is maximizing each of the m_i expressions

$$\sum_{j=1}^{n} z_{i,j,k}^{(s)} \log \psi_{i-1}(r_j; \theta_{i,k}),\ k = 1, \ldots, m_i$$

using Newton-like algorithms to solve

$$\nabla \ell_c = 0 \tag{20}$$

with the partial derivative

$$\frac{\partial \log \phi}{\partial \mu}(x; \mu, \gamma) = -\frac{3}{2x} - \frac{\mu}{x\gamma} + \frac{1}{3\gamma + \mu} + \frac{3\gamma}{2\mu(3\gamma + \mu)} + \frac{\sqrt{\mu}}{2\gamma\sqrt{3\gamma + \mu}} + \frac{\sqrt{3\gamma + \mu}}{2\gamma\sqrt{\mu}}$$

and

$$\frac{\partial \mu_i}{\partial \theta}(\theta) = \frac{\theta}{(1 - u_i)^2} \left(\left(\frac{\theta}{1 - u_i} \right)^2 + \frac{9\gamma_i^2}{4} \right)^{-1/2}$$

Then with

$$\frac{\partial \log \psi_i}{\partial \theta}(x; \theta) = \frac{\partial \mu_i}{\partial \theta}(\theta) \frac{\partial \log \phi}{\partial \mu}(x; \mu_i(\theta), \gamma_i)$$

(20) is equivalently solved by (19) and the solutions of

$$\sum_{j=1}^{n} z_{i,j,k}^{(s)} \frac{\partial \log \psi_{i-1}}{\partial \theta}(r_j; \theta_k) = 0,\ \forall k = 1, \ldots, m_i$$

In order to stop the algorithm, the author in [1] proposes the *Aitken acceleration* to stop the algorithm. The following is directly quoted from [1]:

> *The Aitken acceleration at iteration $s+1$ is given by*
>
> $$a^{(s+1)} = \frac{l^{(s+2)} - l^{(s+1)}}{l^{(s+1)} - l^{(s)}}$$

where $l^{(s)}$ is the observed-data log-likelihood value from iteration s. The estimate of the limit l_∞ of the sequence of log-likelihood values is

$$l_\infty^{(s+2)} = l^{(s+1)} + \frac{l^{(s+2)} - l^{(s+1)}}{1 - a^{(s+1)}}$$

The EM algorithm can be considered to have converged when

$$|l_\infty^{(s+2)} - l_\infty^{(s+1)}| < \epsilon$$

with $\epsilon > 0$ being the desired tolerance.

Finally, we initialize the algorithm with a k-means clustering.

Algorithm 1 EM algorithm for a fixed-priority response times R_i

Require: X-a n-sample of response times of the task τ_i, ϵ-the tolerance of the algorithm, m-the number of cluster to describe X

Ensure: $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}})$

$l_\infty^{(0)} \leftarrow 0$

$l_\infty^{(1)} \leftarrow \epsilon/2$

$s \leftarrow 0$

$\boldsymbol{\mu}^{(0)} = \texttt{kmeans}(X, m)$

$\boldsymbol{\theta}^{(0)} = \left((1-u_i)\sqrt{\mu_k^{(0)}(\mu_k^{(0)} + 3\gamma_{i-1})}, k = 1 \ldots, m \right)$

$\boldsymbol{\pi}^{(0)} = (1/m, \ldots, 1/m)$

while $|l_\infty^{(s+1)} - l_\infty^{(s)}| < \epsilon$ **do**

$\boldsymbol{z}^{(s)} = \left(\frac{\pi_k^{(s)} \psi_{i-1}\left(x_j; \theta_k^{(s)}\right)}{h_i(x_j; \boldsymbol{\pi}^{(s)}, \boldsymbol{\theta}^{(s)})}, j = 1, \ldots, n, k = 1, \ldots, m \right)$ ▷ E-step

$\boldsymbol{\pi}^{(s+1)} = \left(\frac{1}{n} \sum_{j=1}^{n} z_{j,k}^{(s)}, k = 1, \ldots, m \right)$ ▷ M-step

$\boldsymbol{\theta}^{(s+1)} = \operatorname{argmax}_{\theta \in \mathbb{R}_+^m} \ell_{c_2}(\boldsymbol{z}^{(s)}, \theta)$

$a^{(s)} = \frac{l^{(s+1)} - l^{(s)}}{l^{(s)} - l^{(s-1)}}$

$l_\infty^{(s+1)} = l^{(s)} + \frac{l^{(s+1)} - l^{(s)}}{1 - a^{(s)}}$

$(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}}) = (\boldsymbol{\pi}^{(s+1)}, \boldsymbol{\theta}^{(s+1)})$

$s \leftarrow s + 1$

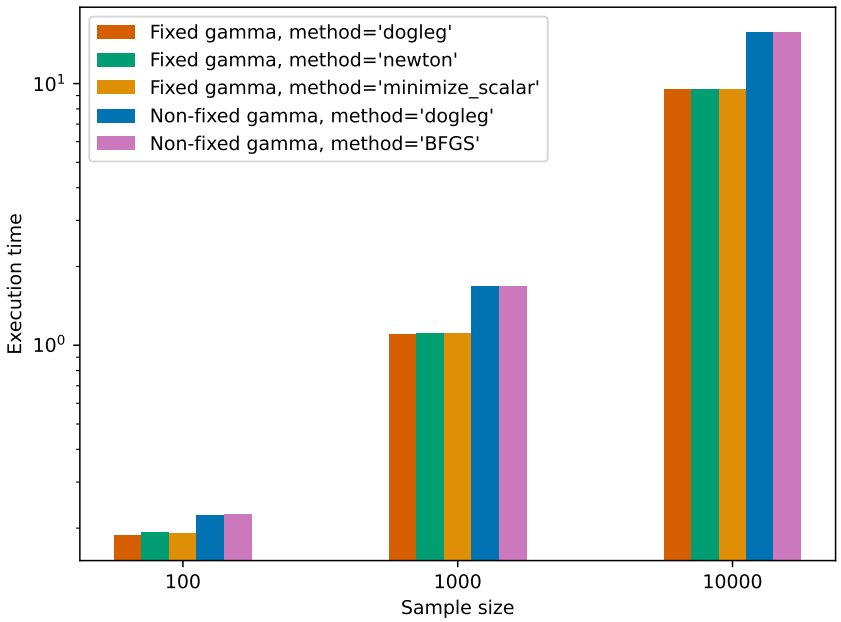


Figure 2: Comparison of several maximization methods. The *non-fixed gamma* model refers to the estimation introduced in [1], the *fixed-gamma* model to the one introduced in Section 4.

4.3 Simulations

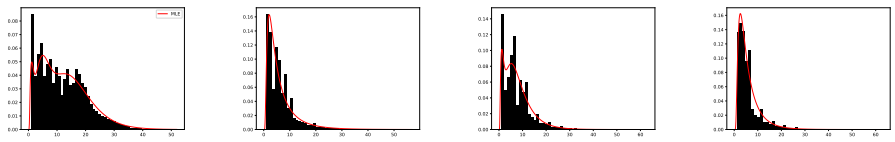
We focus in this section into large task sets. Consider a task set where the $(F_i)_i$ are known and the size of the task set is known to be large enough.

The computation time of the algorithm compared to the method introduced in [1] is shown in Figure 2. The number of parameters is reduced from $2m_i$ to m_i, so the method is expected to be quicker, which it is.

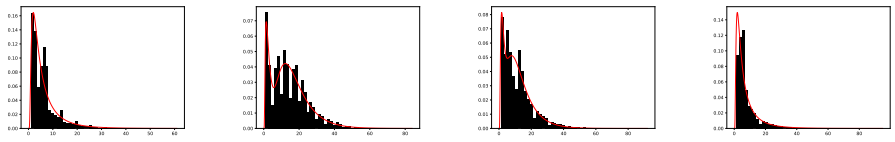
The simulated data is generated using SimSo [2], a framework that is used to generate arrivals of jobs and scheduling policies, that we modified so that execution times and inter-arrival times are probabilistic. In this section, we generate randomly task sets of 40 tasks, and compare the empirical and estimated distributions.

5 Acknowledgments

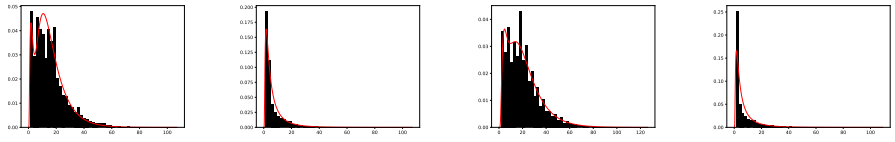
I would like to express my deepest appreciation to Liliana Cucu-Grosjean, Kevin Zagalo, Vincent Runge and Avner Bar-Hen. I am also grateful to all the members of Kopernic team, who inspired me.



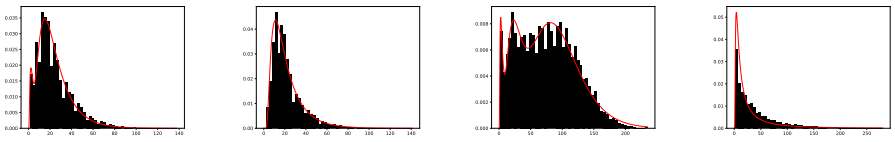
(a) Task 11 (b) Task 12 (c) Task 13 (d) Task 14



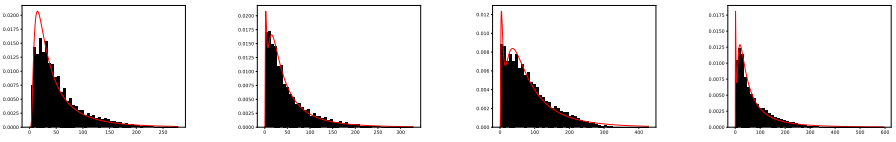
(e) Task 15 (f) Task 16 (g) Task 17 (h) Task 18



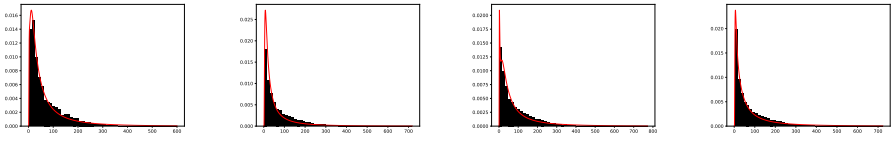
(i) Task 19 (j) Task 20 (k) Task 21 (l) Task 22



(m) Task 23 (n) Task 24 (o) Task 25 (p) Task 26



(q) Task 27 (r) Task 28 (s) Task 29 (t) Task 30



(u) Task 31 (v) Task 32 (w) Task 33 (x) Task 34

Figure 3: Generated response times from SimSo empirical distributions (black) of tasks 11 to 34 and their associated MLE (red).

Bibliography

[1] Punzo A. "A new look at the inverse Gaussian distribution with applications to insurance and economic data". In: *Journal of Applied Statistics* 46.7 (2019), pp. 1260–1287. DOI: 10.1080/02664763.2018.1542668. eprint: https://doi.org/10.1080/02664763.2018.1542668. URL: https://doi.org/10.1080/02664763.2018.1542668.

[2] Maxime Chéramy, Pierre-Emmanuel Hladik, and Anne-Marie Déplanche. "SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms". In: *Proc. of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*. WATERS. 2014.

[3] Kass R. E. and Raftery A. E. "Bayes factors". In: *Journal of American Statistical Association* 90.430 (1995), pp. 773–795.

[4] Schwarz G. "Estimating the Dimension of a Model". In: *Annals of Statistics* 6 (1978), pp. 461–464. DOI: http://dx.doi.org/10.1214/aos/1176344136.

[5] Wu C. F. J. "On the Convergence Properties of the EM Algorithm". In: *The Annals of Statistics* 11.1 (1983), pp. 95–103. URL: https://doi.org/10.1214/aos/1176346060.

[6] Chung Laung Liu and James W Layland. "Scheduling algorithms for multiprogramming in a hard-real-time environment". In: *Journal of the ACM (JACM)* 20.1 (1973), pp. 46–61.

[7] Dempster A. P., Laird N. M., and Rubin D. B. "Maximum likelihood for incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society, Series B* (1977), pp. 1–38.

[8] Kevin Zagalo, Liliana Cucu-Grosjean, and Avner Bar-Hen. "Identification of execution modes for real-time systems using cluster analysis". In: *25th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*. 2020, pp. 1173–1176.

[9] Kevin Zagalo et al. "Response time stochastic analysis for fixed-priority stable real-time systems". In: (2022).